# CS50P W3 - Exceptions

### try and except

x = int(input("What's x? "))

print(f"x is {x}")

\*If user input is a **str**, we will get the runtime error **ValueError** (the value of variable is not the one expected).

1.

* **try**:

x = int(input("What's x? "))

print(f"x is **{**x**}**")

**except** ValueError:

print(“x is not an integer”)

2. Best practice is we should **try** the fewest lines of code possible

* **try**:

x = int(input("What's x? "))

**except** ValueError:

print(“x is not an integer”)

* print(f"x is **{**x**}**")

\*Now if user input is a **str**, we will get the runtime error **NameError:** name ‘**x**’ is not defined. This is because the wrong input triggers the Error handling **`except`** before ‘**x**’ gets to be defined.

3. try and except with **else** block

* **try:**

x = int(input("What's x? "))

**except** ValueError**:**

print(“x is not an integer”)

**else:**

print(f"x is **{**x**}**")

\*Now if the except block does not get triggered, the else block will.

4. try and except with **while** loop and **else** block

* **while** True:

**try:**

x = int(input("What's x? "))

**except** ValueError**:**

print(“x is not an integer”)

**else:**

break

* print(f"x is **{**x**}**")

\*Now, with the while loop, if the except block gets triggered, the program will prompt the user again infinitely until the right value is entered.

5. Create function get\_int()

* **def** main():

x = get\_int()

print(f"x is **{**x**}**")

1. **def** get\_int():

**while** True:

**try:**

x = int(input("What's x? "))

**except** ValueError**:**

print(“x is not an integer”)

**else:**

break

**return** x

main()

1. **def** get\_int():

**while** True:

**try:**

x = int(input("What's x? "))

**except** ValueError**:**

print(“x is not an integer”)

**else:**

**return** x

\*The break is not needed, return x will stop the loop while returning the value.

1. **def** get\_int():

**while** True:

**try:**

x = int(input("What's x? "))

**return** x

**except** ValueError**:**

print(“x is not an integer”)

\*The else block can be removed. Placing return x in the try statement

1. **def** get\_int():

**while** True:

**try:**

**return** int(input("What's x? "))

**except** ValueError**:**

print(“x is not an integer”)

\*We can even avoid an extra line for the return x statement and avoid defining x explicitly by **returning** the result of int(input(...))

6. try and except with **pass**

* **def** get\_int():

**while** True:

**try:**

**return** int(input("What's x? "))

**except** ValueError**:**

pass

\*We can use the pass keyword to not act on the error and simply prompt again.

7. Final refined version of get\_int() adding a function **parameter**

* **def** main():

x = get\_int(“What’s x? “)

print(f"x is **{**x**}**")

1. **def** get\_int(prompt):

**while** True:

**try:**

return int(input(prompt))

**except** ValueError**:**

pass

main()

\*We can improve the implementation of the get\_int() function. It is a good idea for main() to show the prompt argument (“What’s x? ”) and the get\_int() to accept a parameter (prompt).

#### distances.py

**distances** = **{**

"Voyager 1"**:** "163"**,**

"Voyager 2"**:** "136"**,**

"Pioneer 10"**:** "80 AU"**,**

"New Horizons"**:** "58"**,**

"Pioneer 11"**:** "44 AU"

**}**

**def** main():

spacecraft = input("Enter a spacecraft: ")

m = convert(distances[spacecraft])

print(**f**"{m} m away")

**def** convert(au):

return au **\*** 149597870700

main()

\*Because passing a string to the convert() function, the function will attempt to multiply the string by the huge number and this code will result in a **MemoryError**. We need to convert the string before converting the units.

1.

* **def** main():

spacecraft = input("Enter a spacecraft: ")

au = float(distances[spacecraft])

m = convert(au)

print(**f**"{m} m away")

\*Now we fixed the string problem converting the strings to floats before they are passed in as arguments to convert(). But we still have messy data in some of the key-value pairs, where the value is a string composed of letters and digits = **ValueError**

2. **try** and **except**

* **def** main():

spacecraft = input("Enter a spacecraft: ")

**try**:

au = float(distances**[**spacecraft])

**except** ValueError:

print(**f**"Can’t convert ‘**{**distances**[**spacecraft**]}**’ to a float")

return

m = convert(au)

print(**f**"**{**m**}** m away")

\*The try and except block will now handle the ValueError.

3. Adding **except** blocks for different **errors**

* **def** main():

spacecraft = input("Enter a spacecraft: ")

**try**:

au = float(distances**[**spacecraft])

* **except** KeyError:

print(**f**" ‘**{**spacecraft**}**’ is not in dictionary")

return

**except** ValueError:

print(**f**"Can’t convert ‘**{**distances**[**spacecraft**]}**’ to a float")

return

m = convert(au)

print(**f**"**{**m**}** m away")

\*We now have an except block for each ValueErrors and KeyErrors. It is important to try to anticipate potential errors when writing code and include them in the try and except block.