V.1.1

University of Central Lancashire
UCLan

Taster Python Programming Activity

Introduction

Repl.it is an online Integrated Development Environment (IDE) that allows users to write, run, and share code directly from their web browser. It supports a wide range of programming languages and provides a collaborative platform where developers can work together in real-time.

With Replit you can write code and execute code instantly. You can collaborate with other developers which makes it really good for pair programming. Its cloud based so you don't have to worry about setting up the environment on your machine. Everything runs in the cloud! You can learn from others within the community be sharing projects.

Activity

Today's activity will introduce the basics of programming, getting you to build a Python text based game of Rock, Paper, Scissors.

- Your PC should be logged in, if it isn't, please ask the tutor to log you in.
- Visit https://repl.it/ in Chrome.
- To get started you will need to create an account by clicking on "Start Building"
- You will need to provide an email address and password. This means you can continue the experience after your visit today. You will need to verify your account by clicking the verify URL sent to your email address. But you can do this later.

- Click "Create Repl"

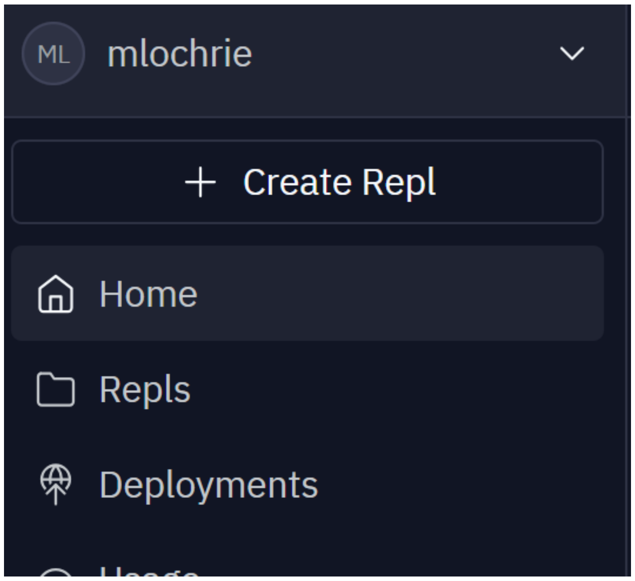
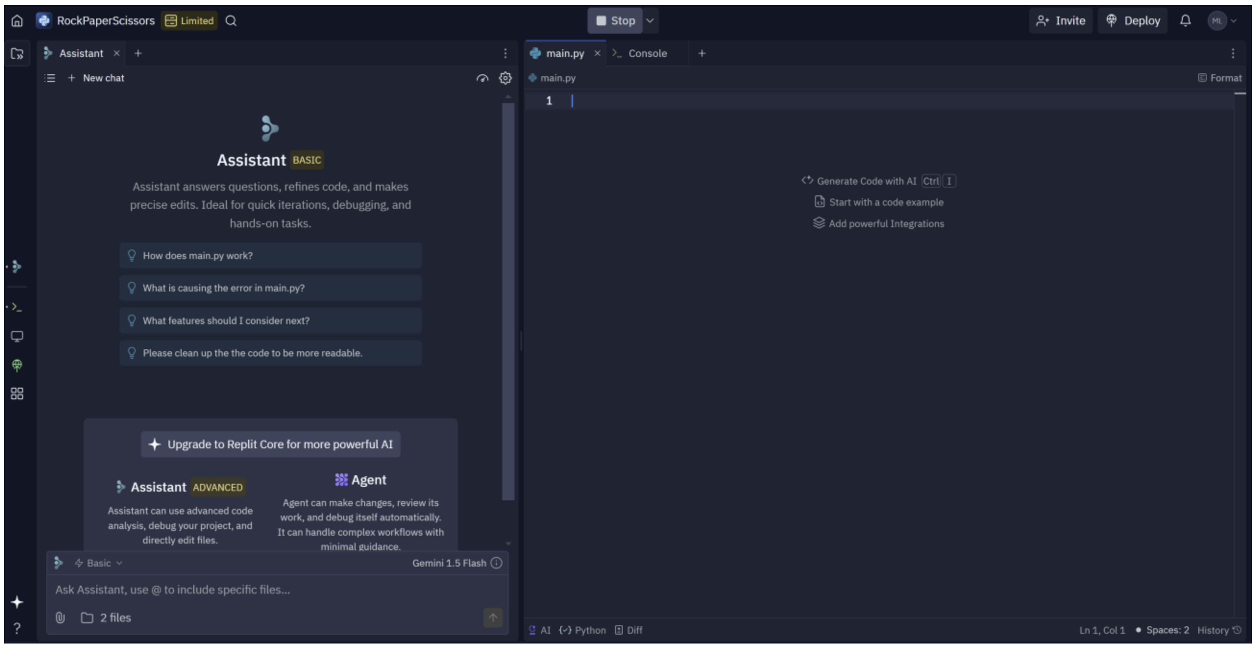


Figure 1. Getting started.

- In the search bar, type python. This is the programming language we are going to use for the taster experience
- Give your project a name "Rock, Paper, Scissors"
- Select "public"
- Create Repl (blue button)
- Once your project is created, you will see two windowpanes. On the left an AI Assistant and on the right your code editor.


RockPaperScissors
Limited
Stop
Invite
Deploy
Assistant
New chat
main.py
Console
Format
Assistant BASIC
How does main.py work?
What is causing the error in main.py?
What features should I consider next?
Please clean up the the code to be more readable.
Upgrade to Replit Core for more powerful AI
Assistant ADVANCED
Agent
Basic
Gemini 1.5 Flash
Ask Assistant, use @ to include specific files...
2 files
Generate Code with AI Ctrl I
Start with a code example
Add powerful Integrations
AI
Python
Diff
Ln 1, Col 1
Spaces: 2
History

Assistant answers questions, refines code, and makes precise edits. Ideal for quick iterations, debugging, and hands-on tasks.

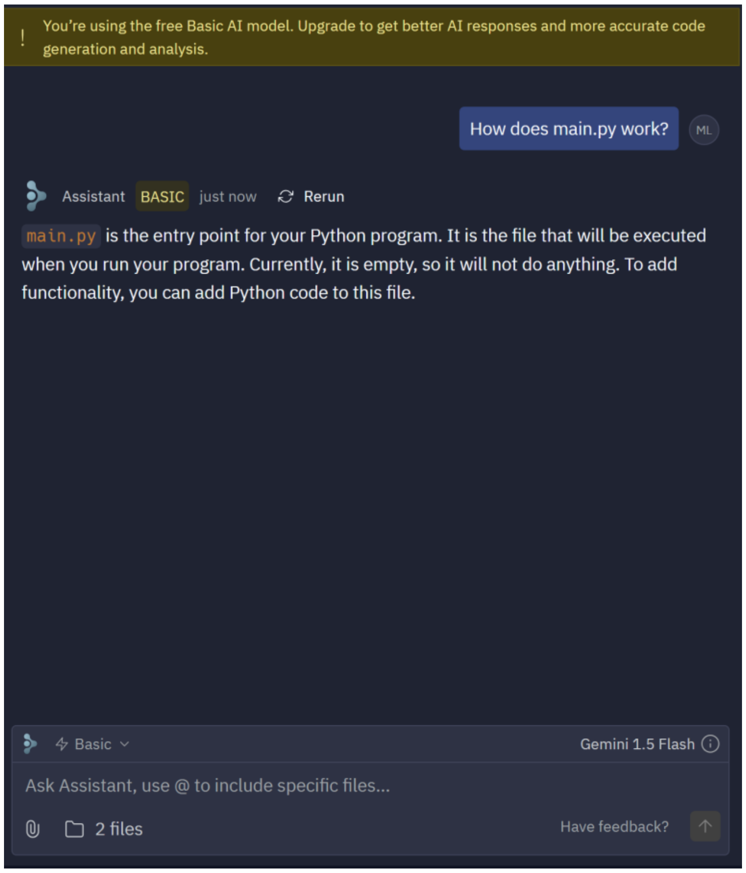
Assistant can use advanced code analysis, debug your project, and directly edit files.

Agent can make changes, review its work, and debug itself automatically. It can handle complex workflows with minimal guidance.

Figure 2. Running the code.

  - You will be writing the code in the code editor pane. The AI Assistant is there to help you through the development process.

- Before we begin, lets try out the AI Assistant, click on the prompt "How does main.py work" and this will add the prompt to the textbox, click the send button. This will add the prompt and perform the query. The AI Assistant will then respond explain the purpose of the main.py


You're using the free Basic AI model. Upgrade to get better AI responses and more accurate code generation and analysis.

How does main.py work?

Assistant BASIC just now Rerun

`main.py` is the entry point for your Python program. It is the file that will be executed when you run your program. Currently, it is empty, so it will not do anything. To add functionality, you can add Python code to this file.

Figure 3. AI Assistant

- Moving on we will start to write our code for the game.

Rock, Paper, Scissors: Technical Specification

You are to build the rock, paper, scissors game where you will need to think about the logic involved: how the game is won, lost, and drawn.

The player will start by typing their choice, the computer will randomly select theirs and the application will determine who wins!

You will learn about various programming constructs: statements, loops, variables etc.

Use the AI Assistant and lab tutor to help you along the way

Step 1: The Foundations

- Import the packages we need to support out development - As we will be creating a game for User vs Computer, we need to add an element of random to our program.

The "random" library will be used for as functionality for generating random numbers. Which will be used to simulate the computer's choice in the game.

```
import random
```

- Using the AI Assisant, lets ask it how the code works. Type in the prompt "how does the code work?" and run this prompt.
    - You should get a response similar to "The code imports the random module. This module provides functions for generating random numbers. You can use it to generate random integers, floats, and choices from a list."
- Moving on we are going to build out the game. The below code contains the main function which is used as the entry point of our game and the play_game function is where all our logic will go. When writing code in Python be careful with where you place code (note the print statement is tabbed in once). This means the print statement will be executed when the play_game function is called.

```
def play_game():
  print("This is the start of our game")


if __name__ == "__main__":
  play_game()
```

- Inside the play_game function we need to create some variables to store initial values (remember to tab in the variables)

```
choices = ["rock", "paper", "scissors"]
user_score = 0
computer_score = 0
```

  - You can test the code by clicking "Run"
- At the moment it won't do much, but you should see the message "This is the start of our game".

Step 2: Building the logic

- You are going to make the game run until stopped by the user. To do this you will need to wrap all your login inside a loop.
- Add the following code to the play_game function.

```
while True:
```

- We will first get the computer to decide on their choice
    - Write computer_choice = random.choice(choices)
        - This will use the array of choices we have declared previously, select on at random and store it inside the variable computer_choice
- Next we will ask the user to type their choice

  - Write user_choice = input(“Enter rock, paper, scissors (or ‘exit’ to quit): ”).lower()
- The next block of code is used to detect if the user enters exit and if so terminates the game. The code also checks to see if a non option was chosen and if so an error message is presented to the user.

```
if user_choice == 'exit':
      break

if user_choice not in choices:
       print("Invalid choice. Please try again.")
      continue
```

- Moving on, print the selections from the computer and user using the print command.
- Now for the logic to determine a winner.
  - We are going to use an if/else statement approach
- First check to see if the choices are the same

```
if user_choice == computer_choice:
      print("It's a tie!")
```

- Then write the else/if statement and check to see if the user chooses “rock” and the computer chooses “scissors”

```
elif (user_choice == "rock" and computer_choice == "scissors") or \
```

- Expand these checks for paper and scissors from the user’s choice. You can do this by entering this code on the next line

```
(user_choice == "paper" and computer_choice == "rock") or \
```

- Complete the logic for scissors.
- At the end of this statement add a print statement saying “You win”
- And update the users score by 1

```
user_score += 1
```

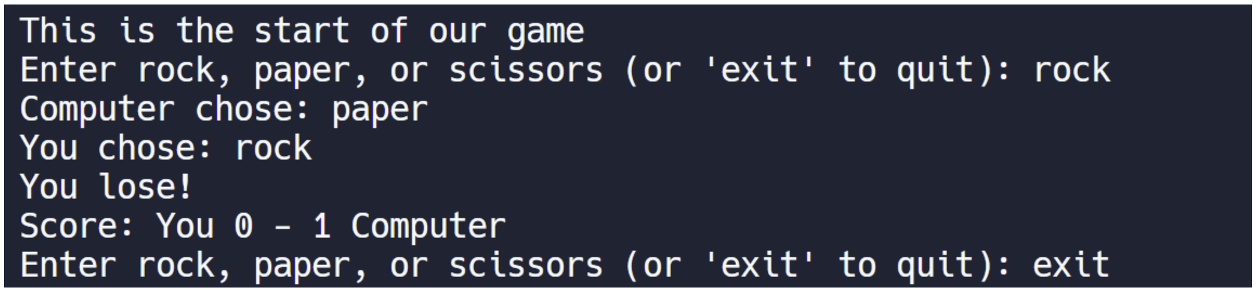
- Next you are going to write the else statement for when the computer wins

```
else:
       print("You lose!")
       computer_score += 1
```

- Below the else statement print the score for the user and computer

Step 3: Testing the game

- Run the code and give your game a playtest



```
This is the start of our game
Enter rock, paper, or scissors (or 'exit' to quit): rock
Computer chose: paper
You chose: rock
You lose!
Score: You 0 - 1 Computer
Enter rock, paper, or scissors (or 'exit' to quit): exit
```

Figure 4. Screenshot of the game with user input

- To end the game type exit, work out where a print statement should be added so that when the user quits the game a nice goodbye message is displayed.

Well done! – if you want to continue your progress in your own time, we'd love to hear from you StudyComputing@uclan.ac.uk

Completed code https://mlochrie.github.io/recruitment/taster/python/rock-paper-scissors.py

Code Explanation

Python is a popular high-level programming language known for its readability and versatility. It is used in a wide range of applications, including:

- Web Development: Python frameworks like Django and Flask are used to build web applications.
- Data Science and Machine Learning: Python's libraries like NumPy, Pandas, Scikit-learn, and TensorFlow make it a powerful tool for data analysis and machine learning.
- Scripting and Automation: Python is used for automating tasks, such as system administration and web scraping.
- Game Development: Python can be used to create games, particularly with libraries like Pygame.

Concepts:

- **Functions:** The code defines a function called play_game(). This function encapsulates the logic for playing the game. This allows for code organization and reusability.
- **Loops:** The while True: loop runs indefinitely until the user chooses to exit by entering 'exit'. This loop allows the game to continue until the user decides to stop.
- **Conditional statements:** The if, elif, and else statements are used to determine the outcome of each round of the game. The code checks the user's choice and the computer's choice to decide who wins or if there is a tie.
- **Input and Output:** The input() function allows the user to enter their choice. The print() function displays messages to the user, including the computer's choice, the result of the round, and the current score.
- **Randomness:** The random.choice() function is used to select the computer's choice randomly from the list of choices. This ensures a fair and unpredictable game.
- **Variables:** The code uses variables to store the user's choice, the computer's choice, the user's score, and the computer's score. These variables allow the code to track the state of the game and update it as the game progresses.

Key Elements:

The key elements of this code are:

- Function Definition: The code defines a function called play_game(), which encapsulates the logic for playing the game. This allows for code organization and reusability.
- Loop: The while True: loop runs indefinitely until the user chooses to exit by entering 'exit'. This loop allows the game to continue until the user decides to stop.
- Conditional Statements: The if, elif, and else statements are used to determine the outcome of each round of the game. The code checks the user's choice and the computer's choice to decide who wins or if there is a tie.

- Input and Output: The input() function allows the user to enter their choice. The print() function displays messages to the user, including the computer's choice, the result of the round, and the current score.
- Randomness: The random.choice() function is used to select the computer's choice randomly from the list of choices. This ensures a fair and unpredictable game.
- Variables: The code uses variables to store the user's choice, the computer's choice, the user's score, and the computer's score. These variables allow the code to track the state of the game and update it as the game progresses.
- its nuances and become proficient in writing and understanding its code.

Key Points for the activity:
The key learning is to understand how to use functions, loops, conditional statements, input and output, randomness, and variables to create an interactive game. This code demonstrates how to combine these concepts to build a simple Rock Paper Scissors game against a computer.

V.1.1

Code

```
import random

def play_game():
  print("This is the start of our game")





  choices = ["rock", "paper", "scissors"]
  user_score = 0
  computer_score = 0




  while True:



    computer_choice = random.choice(choices)
```

Explanation

Imports:
This line imports the random module, which provides functions for generating random numbers. You will use the random.choice() function to randomly select the computer's choice.

Functions:
This line defines a function called play_game(). This function encapsulates the logic for playing the game. This allows for code organization and reusability.

Variables:
This line creates a list called choices that contains the three possible choices for the game: "rock", "paper", and "scissors".
This line initializes a variable called user_score to 0. This variable will keep track of the user's score throughout the game.
This line initializes a variable called computer_score to 0. This variable will keep track of the computer's score throughout the game.

Mechanics:
starts a while True: loop. This loop will run indefinitely until the user chooses to exit.

randomly selects one of the choices from the choices list and assigns it to the computer_choice variable. This is how the computer chooses its move.

```
    user_choice = input(
        "Enter rock, paper, or scissors (or 'exit' to
quit): ").lower()

    if user_choice == 'exit':
      break

    if user_choice not in choices:
      print("Invalid choice. Please try again.")
      continue

    print(f"Computer chose: {computer_choice}")
    print(f"You chose: {user_choice}")

    if user_choice == computer_choice:
      print("It's a tie!")
    elif (user_choice == "rock" and computer_choice ==
"scissors") or \
         (user_choice == "paper" and computer_choice ==
"rock") or \
         (user_choice == "scissors" and computer_choice
== "paper"):
      print("You win!")
      user_score += 1
    else:
```

prompts the user to enter their choice. The input is converted to lowercase using .lower() to make the comparison case-insensitive.

checks if the user entered 'exit'. If they did, the loop breaks, ending the game.

checks if the user's input is a valid choice. If not, it prints an error message and continues to the next iteration of the loop, prompting the user again.

checks if the user and computer chose the same option. If they did, it prints "It's a tie!".

checks if the user won the round based on the rules of Rock Paper Scissors. If they did, it prints "You win!" and increments the user's score by 1.

V.1.1

```
      print("You lose!")
      computer_score += 1

    print(f"Score: You {user_score} - {computer_score}
Computer")

  print("Thanks for playing!")



if __name__ == "__main__":
  play_game()
```

play_game() function is only called when the script is run directly. This is a standard practice in Python to prevent the function from being called unintentionally when the script is imported as a module.